Better Analysis of Defect Data at NASA

Tim Menzies[1], Robyn Lutz[2,3], Carmen Mikulski[2]

[1] Lane Department of Computer Science, West Virginia University
    tim@menzies.com
[2] Jet Propulsion Laboratory, California Institute of Technology
    robyn.r.lutz | Carmen.Mikulski@jpl.nasa.gov
[3] Department of Computer Science, Iowa State University

WP ref: 02/superodc/superodc. July 18, 2002

1 Introduction

A fool learns from her own mistakes;
the wise learns from the mistakes of others.
— Anonymous

This quote illustrates the goal of defect analysis at NASA. Based on past experience, missions should be run in a better way that avoids the mistakes of the past while continually improving future practice.

Much recent progress has been made towards this goal. Lutz and Mikulski (hereafter, the LM team) recently completed an extensive study of 191 ISA (Incident/Surprise/Anomaly) reports on deep space NASA missions [2,3]. The challenge with analyzing those reports was that they were hard to analyze since they were expressed in loosely-structured free text. Hence, the LM team applied IBM's *orthogonal defect classification* (ODC [1]) technique to the text of those messages. ODC is a structuring tool for defect reports. The ODC terms used in this study are shown in Figure 1:

— New defects are tagged with the *activity*-in-progress and *triggering* event that revealed the problem;
— When defects are fixed, analysts record the *target* of the fix and the defect *type*.

Fig. 1 Defect classifications used in the study. Right-hand-side terms are either the *activity's triggers* or the *target's type*.

This simple categorization scheme injected enough structure into the anomaly reports to permit some data mining. The LM team found five clusters of *activities-triggers-targets-types* that reflect repeated patterns in defect incidents at NASA.

These clusters where then explored in detail. The LM team tried to explain why each cluster existed. These explanations were then assessed with the assistance of JPL mission experts. As a result of these reviews, one of the explanations was confirmed, two were rejected, and the remaining three were modified. 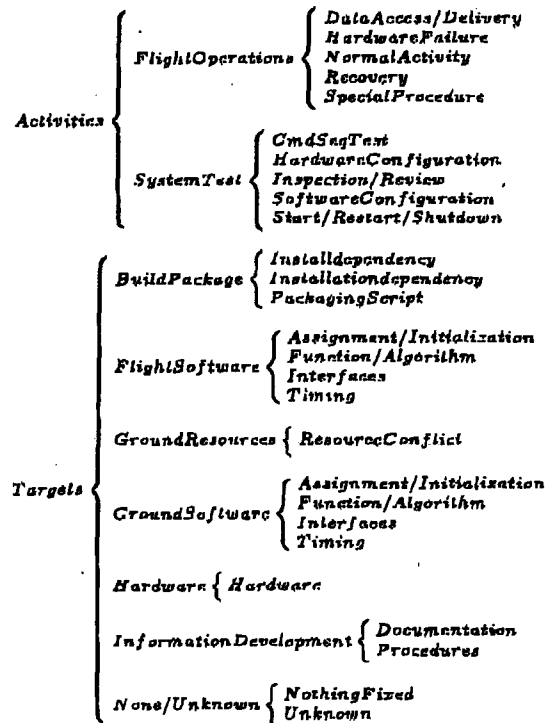These confirmed and modified explanations were then used to derive recommendations for changes to NASA's operational guidelines for deep space missions.

This experiment was successful enough to produce the following results:

1. The LM team now works with a spacecraft project currently in the testing phase to apply ODC.
2. The *JPL problem-reporting team* is discussing support for ODC classification in their next-generation toolset

Target	Trigger											Total
	Cmd Seq Test	Data Access/ Delivery	H/W Configuration	H/W Failure	Inspection/ Review	Normal Activity	Recovery	S/W Configuration	Special Procedure	Start/ Restart/ Shutdown	Unknown	
Build Package	4							4				8
Flight S/W	2	5	2	6	4	3	12	1	7			42
Ground Resources				1		1		1				3
Ground S/W	6	20	3		3	2		6	2	4		46
Hardware				4	2							6
Info. Development		29	3	3	3	8	2	6	5		3	62
None/Unknown		15	2	3	2	3	2	2	2		4	35
Totals	12	69	10	17	14	17	16	20	16	4	7	

Fig. 2 ODC trigger and types seen in JPL applications.

3. The *IV&V defect tracking team* at NASA's Independent Verification and Validation facility is configuring a commercial defect tracing tool to support ODC. Their plan is to purchase many licenses and give these away to projects, in return for their defect data.

Consequently, in the near future, we can expect a very large number of ODC datasets from a wide range of NASA projects. Hence, two questions now become relevant:

1. Can current ODC analysis methods scale up to handle a very large number of datasets?
2. If not, how can we optimize the ODC analysis?

In summary, our argument will be that automatic *machine learning* methods can simplify part of the ODC analysis. Experiments with this approach have shown that human expertise can out-perform automatic machine learning. However, in two cases, the combination of ODC plus machine learning has significant productivity advantages. Those cases are:

- When the data set is too large or too complex for humans to explore manually.
- When the required domain expertise is in short supply

2 Manual ODC Analysis

This section reviews the strengths and weaknesses of current ODC analysis. Recall that that process consists of five steps:

1. *Initial categorization:* When anomalies are detected, activities and triggers are recorded.
2. *Final categorization:* When anomalies are resolved, targets and types are recorded.
3. *Clustering:* Once the anomalies are classified, the are clustered.
4. *Explanation:* Once they clusters are identified, they are reviewed. Review comprised three steps: explanation generation, explanation review, explanation modification. During modification, explanations can be rejected.
5. *Action plan:* Once the set of explanations are stable, they can be used to motivated changes to current practices.

In the initial study of the LM team, the *initial* and *final categorization* was a manual process of loosely-structured English defect reports. This process is time-consuming and will be avoidable once the JPL problem-reporting tool and the IV&V defect tracking tool are modified to include the ODC categories. That is, once the data entry method has been

modified to an ODC framework, no post-processing will be required to categorize the defect reports.

Clustering is a process of looking for patterns of ODC classifications that often occur together. The LM team used the standard spreadsheet program to find high frequency patterns. For example Figure 2 shows one generated spreadsheet. Three features of interest are:

1. Distribution within classifications is uneven. Research should focus on the small number of high-frequency problems.
2. The large number of anomalies seen during sending commands and receiving data from spacecraft (see the large numbers in the *data access/delivery* column).
3. The large number of anomalies in the *information Development* and *ground software development* cells. When the LM team checked these patterns, they discovered that the receipt of information from the spacecraft by the ground systems was subject to many technical complications.

While a simple spreadsheet analysis is adequate for the LM team, other studies might require more elaborate tools. For example, the LM team only used five of the eight dimensions of the standard ODC scheme (the four shown in Figure 1 plus "impact", which is implicit in the selection by the LM team of only high-criticality anomalies). The other three dimensions were not used due to domain-specific reasons, e.g. they were invariant in these projects. These domain-specific reasons for restricting the dimensionality may not apply to other domains. For example:

- The IV&V defect tracking team is planning to collect ODC-type data on defects seen in NASA applications as well as a range of static code metrics such as parameters describing the data flow and control flow *structures* of a program.
- In all, it is anticipated that these data sets will contain not just the four dimensions studied by the LM team, but up to 40 dimensions.
- It is unlikely, to say the least, that simple spreadsheet methods will suffice for studying such a complex space.

Once the clusters are generated, they must be audited. The LM team audited the clusters by trying to generate *explanations* for each one. This is an inherently manual process since it requires a detailed discussion with business users about the significance of each cluster.

Similarly, generating the *action plan* is also an inherently manual process. Action plans reflect changes to current practices and if such changes are not endorsed by practitioners, they may be ignored. Hence, it is vital to carefully discuss the explanations with the user community in order to determine the most clearly motivated actions.

In summary, of the five steps in LM-style ODC, the last two should not be automated further and automatic support tools are being already coded for the first two steps. The remaining step of clustering is a clearly a candidate for automation since, in the case of datasets with large dimensionality, automatic support will be essential to scaling up the technique.

3 Automatic Clustering via Machine Learning

Many of the numerous clustering algorithms currently available make some assumption of the parameters being clustered. For example, a common assumption is that the parameters come from a simple linear continuous function. While this assumption simplifies methods such as (e.g.) regression, they may not be appropriate for discrete data. Discrete data sets can contain "holes" where the target function changes radically. Two alternatives to continuous clustering methods are *association rule learning* that seeks patterns of attributes [6] and *treatment learners* that find control actions that select for preferred attributes [4].

3.1 Experiments with Association Rule Learners

Association rule learners make no assumption of linearity and so can succeed when (e.g.) regression fails. In the association $LHS \implies RHS$, no attribute can appear on both sides of the association; i.e. $LHS \cap RHS = \emptyset$. The rule $LHS \implies RHS$ holds in the example set with *confidence* c if $c\%$ of the examples that contain LHS also contain RHS; i.e. $c = \frac{|LHS \cup RHS|}{|LHS|}$. The rule $LHS \implies RHS$ has *support* s in the example set if $s\%$ of the examples contain $LHS \cup RHS$; i.e. $s = \frac{|LHS \cup RHS|}{|D|}$ where $|D|$ is the number of examples. Association rule learners return rules with high confidence (e.g. $c > 90\%$). The search for associations is often culled via first rejecting associations with low support.

For example, in the following example set with five items:

1. Bread, milk
2. Beer, diaper, bread, eggs
3. Beer, coke, diaper, milk,
4. Beer, bread, diaper, milk,
5. Coke, bread, diaper, milk

an association rule learner could find association

$$Diaper\ AND\ Beer \implies Milk@66$$

i.e. the rule has a confidence of $\frac{2}{3} = 0.66$. When applied to the ODC data used by the LM team, the following associations were discovered:

1. *Trigger=DataAccess/Delivery* $\implies$
   Activity=FlightOperations @1
2. *Type=Procedures* $\implies$ *Target=InformationDevelopment @1*

3. *Activity=FlightOperations AND
   Target=InformationDevelopment* $\implies$
   Type=Procedures @0.96
4. *Activity=FlightOperations AND Type=Procedures* $\implies$
   Target=InformationDevelopmen @1
5. *Target=InformationDevelopment 62* $\implies$ *Type=Procedures @0.95*

Note that these were all of high confidence ($> 90\%$). Most of these associations are not informative; e.g. association 1 and association 5 just report that part of the hierarchy shown in Figure 1 appears in the data. However, in terms of the theme of this paper, associations 3 is very interesting. Association 3 reports a major clustering of defects around *flight operations, information development, and procedures*. The same pattern was a major finding of the LM team. That is, this automatic method can reproduce at least some of known results of a manual process.

What is also interesting about these associations is what is *not* appearing. Recall from Figure 1 that there are *flight operations* activity has five *triggers*: *data access/delivery, hardware failure, normal activity recovery* and *special procedure*. Only *one* of these triggers appears in the associations; i.e. *data access/delivery*. This is surprising since a pre-experimental intuition was that triggers such as *hardware failure* and *recovery* would be a major source for defects. This counter-intuitive result is exactly the kind of result we seek from defect analysis since it tells us what intuitions to revise and where to best focus future effort.

Another missing feature of the above associations is that these form only one of the five clusters identified by the LM team. Clearly, a manual analysis by experts within an area can reveal more than an automatic analysis by a machine learner that can only access the symbols within a data set. However, in cases where the data set is too complex for manual analysis, or in circumstances when the experts are not available, it is useful to know that at least something can be achieved via automatic methods.

3.2 Experiments with Treatment Learners

The TAR2 *treatment learning* assumes that each class has a numeric value representing its worth to a user. For this study, we went back to the ODC logs and found that each defect was scored according to a criticality scale "*A..E*", where "*A*" was highest priority. A *treatment* is a constraint that selects for a *more interesting* ratio of classes; e.g. one with a larger proportion of criticality "*A*" anomaly reports. For example, it is possible to ask TAR2 "what is the constraint that *most* selects for the highest/lowest criticality errors?". By "most" we mean that when TAR2 terminates, it has searched all combinations of possible constraints to find the most influential ones. Theoretically, this search takes exponential time and is intractable. In practice, the algorithm works successfully (a curious phenomenon explored elsewhere [5]).

TAR2's results are best viewed as a comparison between a *baseline* distribution and the distribution of classes seen after applying the learnt constraint. For example, the *baseline* plot
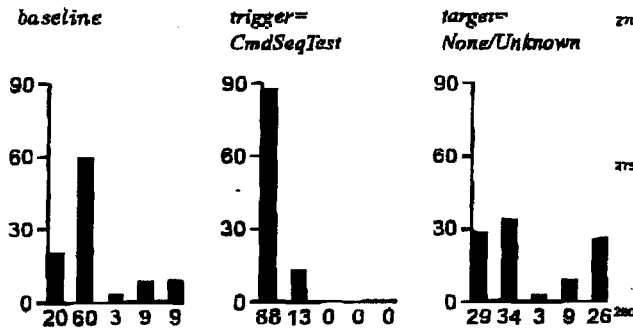
Fig. 3 Distribution of criticality one through five seen in this study. Each histogram is sorted left-to-right highest to lowest criticality.

of Figure 3 shows the ratio of criticality "A" through criticality "E" anomalies in the ODC data set.

TAR2 found that the constraint that most selects for criticality "A" anomalies was *trigger= command sequence test*. Note that the imposition of this constraint shifts the ratio of criticality classes to the left-hand-side of the middle histogram of Figure 3; i.e. it selects for *more* high criticality anomalies. Command sequence tests selecting for high criticality anomalies is hardly surprising. These tests are a activity that are designed to guide changes to the software prior to operations.

TAR2 also found that the constraint that most selects for criticality "E" anomalies was *targets= none/unknown*. Note that the imposition of this constraint shifts the ratio of criticality classes to the right-hand-side of the right histogram of Figure 3; i.e. it selects for *fewer* high criticality anomalies. *None/unknown* was a category for anomalies where no fix was made (e.g., the reported behavior was in fact, acceptable, or the anomalous scenario could not recur during the rest of the mission) or where the fix was not described (inadequate documentation). At first glance, there is no business interpretation for this treatment. However, consider the case where treatment learning was applied to incrementally applied to anomaly data during the life time of that project. If this category starts appearing as a significant treatment, then that would suggest that the data collection instrument should be improved. That is, the prominence of this category could serve as a flag warning the QA team that, e.g., corrective actions were not being fully described.

4 Discussion

We have argued for the need to better automate the clustering part of an ODC analysis. Are association rule learning and treatment learning good candidates for this automation support? Based on the above experiment, we offer several conclusions.

A manual analysis by experts is preferable to automatic machine learning. Automatic machine learners can only access the symbols in a training set. Human experts skilled in the domain being studied can access more background knowledge about a domain. Hence, they can generate better theories and,

in the case of an LM-style ODC analysis, more explanations for the clusters in the data than any machine learner. However, for data sets with large dimensionality, arguments for the superiority of manual clustering analysis by human experts are spurious. Manual clustering methods using simple spreadsheet methods work when dealing with five attributes. These methods may not scale to the data sets we expect to see in the near future. Automatic machine learning, on the other hand, does scale to very large data sets and data sets with large dimensionality.

There are two other cases where machine learning might be preferred to the mostly-manual analysis of the LM team:

- The audit check of TAR2 described above (where a "catch-all" activity appeared in a treatment) could be applied incrementally whenever anomaly data sets are updated. Such audit checks would ensure the quality of the data, as it is collected. Note that automatic analysis is faster than an automatic method; i.e. incrementally applying machine learners to check for data quality would be a cheap activity.

- In the case where the experts are unavailable, machine learning methods can reproduce at least some of the analysis of human experts. Hence, ODC coupled with machine learning could scale to more sites across the NASA enterprise.

Availability of Software

The TAR2 treatment learner is available from http://www.ece.ubc.ca/twiki/bin/view/Softeng/TreatmentLearner. The APRIORI association rule learner is available within the WEKA toolkit [7] from http://www.cs.waikato.ac.nz/~ml/weka/. WEKA is a JAVA tool and runs on most platforms. TAR2 is compiled for Windows but is simple to compile for UNIX. Both WEKA and TAR2 are freely available open source tools.

Acknowledgements

References

1. R. Chillarege, I. S. Bhandari, J. K. Chaar, M. J. Halliday, D. S. Moebus, B. K. Ray, and M.-Y. Wong. Orthogonal defect classification—a concept for in-process measurements. *IEEE Transactions on Software Engineering*, 18(11):943–956, November 1992.

2. R. Lutz and C. Mikulski. Empirical analysis of safety-critical anomalies during operations, 2003. (submitted).

3. R. Lutz and C. Mikulski. Operational anomalies as a cause of safety-critical requirements evolution. *Journal of Systems and Software (to appear)*, 2003. Available from http://www.cs.iastate.edu/~rlutz/publications/JSS02.ps.

4. T. Menzies, E. Chiang, M. Feather, Y. Hu, and J. Kiper. Condensing uncertainty via incremental treatment learning. In *Annals of Software Engineering (submitted)*, 2002. Available from http://tim.menzies.com/pdf/02itar2.pdf.

5. T. Menzies and Y. Hu. Just enough learning (of association rules). In *WVU CSEE tech report*, 2002. Available from http://tim.menzies.com/pdf/02tar2.pdf.

6. R.Agrawal, T.Imeilinski, and A.Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD Conference, Washington DC, USA*, 1993. Available from http://citeseer.nj.nec.com/agrawal93mining.html.

7. I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.

Back to SEL Home Page

27th Annual IEEE/NASA Software Engineering Workshop

December 4-6, 2002
Greenbelt Marriott Hotel, Greenbelt, MD, USA
Sponsored by NASA Goddard Space Flight Center, Software Engineering Laboratory;
        IEEE Computer Society, Technical Council on Software Engineering

Background:
The 27th Annual IEEE/NASA Software Engineering Workshop will be held at the
Greenbelt Marriott Hotel, Greenbelt, Maryland, in Metropolitan Washington DC,
December 5-6, 2002, following a day of tutorials on December 4. The workshop will
be co-located with IEEE ICECCS 2002, the 8th IEEE International Conference on
Engineering of Complex Computer Systems (2-3 December). ICECCS attendees may attend
the tutorials on December 4, but ICECCS has a separate paper submission policy, and
ICECCS papers will be reviewed by a separate Program Committee. A discount will be
offered to persons attending both events.

Scope:
The workshop aims to bring together NASA technical staff, contractors, academics
and industrial practitioners interested in the advancement of software engineering
principles and techniques. The workshop provides a forum for reporting on past
experiences for describing new and emerging results and techniques, and for
exchanging ideas on best practice and future directions. Of particular importance
is relevance to NASA+IBk-s mission and goals, and how techniques might be applied,
or adapted for use, at NASA, or how NASA+IBk-s techniques might be used or adapted
for more generic use.

Topics of interest include, but are not limited to:
- Metrics and experience reports
- Software quality assurance
- Formal methods and formal approaches to software development
- Software engineering processes and process improvement
- CMM and CMMI
- Requirements engineering
- Software Architectures
- Real-time Software Engineering
- Software maintenance, reuse, and legacy systems
- Agent-based software systems

Submissions:

*** PLEASE NOTE NEW DETAILS ***

Extended abstracts (4 to 6 pages) should be submitted electronically (plain-text,
.doc, .pdf or .ps formats only, please) to: sew27@listserv.gsfc.nasa.gov by ** 9
August 2002 **.

If you already have a full paper written, you may submit that instead of an
abstract.
If you do not receive an acknowledgement for your submission within 3 working days
(or have previously submitted and not received an acknowledgement) it may have been
the victim of our firewall.

Please resend your submission to: mike.hinchey@usa.net and it will be acknowledged
asap.
Thank you.

Abstracts will be reviewed by the Program Committee, and full versions of selected papers will be due one week after the workshop (13 December) to be included in the workshop proceedings that will be published by IEEE Computer Society Press. At least one author per paper will be required to register and present at the workshop.

Any inquiries should also be directed by email to: sew27@listserv.gsfc.nasa.gov

Conference Organizers:
Conference Chair: Mike Hinchey
Program Committee Chair: Margaret Caulfield
Finance Chair: Don Jamison
Registration Chair: John Cook
Tutorials Chair: Mike Stark

Back to SEL Home Page

See the NASA Website Privacy Statement and the
NASA Information Technology (IT) Security Warning.